



日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 1 1 月 1 7 日
Date of Application:

出 願 番 号 特 願 2 0 0 3 - 3 8 6 0 9 1
Application Number:
[ST. 10/C]: [J P 2 0 0 3 - 3 8 6 0 9 1]

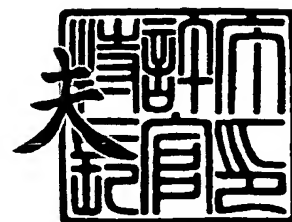
出 願 人 株式会社日立製作所
Applicant(s):

*U.S. Appln. Filed 1-23-04
Inventor: S. Oshima et al
mattingly Stanger & Malur
Docket ASA-1161*

2 0 0 4 年 1 月 8 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康



出証番号 出証特 2 0 0 3 - 3 1 0 9 6 8 6

【書類名】 特許願
【整理番号】 K03007621A
【あて先】 特許庁長官殿
【国際特許分類】 G06F 9/00
【発明者】
 【住所又は居所】 神奈川県川崎市麻生区王禅寺 1 0 9 9 番地 株式会社日立製作所
 システム開発研究所内
 【氏名】 大島 訓
【発明者】
 【住所又は居所】 愛知県尾張旭市晴丘町池上 1 番地 株式会社日立製作所情報機器
 事業部内
 【氏名】 富田 理
【発明者】
 【住所又は居所】 神奈川県川崎市麻生区王禅寺 1 0 9 9 番地 株式会社日立製作所
 システム開発研究所内
 【氏名】 木村 信二
【特許出願人】
 【識別番号】 000005108
 【氏名又は名称】 株式会社 日立製作所
【代理人】
 【識別番号】 100075096
 【弁理士】
 【氏名又は名称】 作田 康夫
【選任した代理人】
 【識別番号】 100100310
 【弁理士】
 【氏名又は名称】 井上 学
【手数料の表示】
 【予納台帳番号】 013088
 【納付金額】 21,000円
【提出物件の目録】
 【物件名】 特許請求の範囲 1
 【物件名】 明細書 1
 【物件名】 図面 1
 【物件名】 要約書 1

【書類名】 特許請求の範囲**【請求項 1】**

アプリケーションプログラムと共有ライブラリとを同一の仮想空間に配置し、かつ、アプリケーション毎に固有の仮想空間を有するようメモリ管理を行うオペレーティングシステムを実行する計算機において、

前記アプリケーションによる前記共有ライブラリへの関数呼出を注入共有ライブラリによってフックし、

前記注入共有ライブラリが前記メモリのカーネル領域に格納されたモジュールを呼び出し、

前記モジュールに設定された情報に基づいて該モジュールが処理を行い、その後、前記モジュールが前記共有ライブラリの関数を呼び出すことを特徴とする共有ライブラリ呼び出し方法。

【請求項 2】

前記モジュールが、

前記フックされた前記共有ライブラリへの呼出の内容を前記カーネル領域内に記録することを特徴とする請求項 1 記載の共有ライブラリ呼び出し方法。

【請求項 3】

前記モジュールが、

前記呼出の内容をあらかじめ定められたポリシーに照らして監査し、

前記共有ライブラリの呼出の可否を決定することを特徴とする請求項 2 記載の共有ライブラリ呼び出し方法。

【請求項 4】

前記呼び出しの可否の結果を前記カーネル領域内に記録することを特徴とする請求項 3 記載の共有ライブラリ呼び出し方法。

【請求項 5】

前記呼出内容を前記カーネル領域内に記録する領域をプロセッサ毎に設けることを特徴とする請求項 4 記載の共有ライブラリ呼び出し方法。

【請求項 6】

前記カーネル領域内に記録された呼出内容を取得する出力アプリケーションを有することを特徴とする請求項 5 記載の共有ライブラリ呼び出し方法。

【請求項 7】

前記出力アプリケーションが前記モジュールの設定を変更するコマンドを発行し、

前記コマンドを、前記カーネル領域の所定の領域に格納し、

前記モジュールが前記所定の領域に格納されたコマンドに基づいて設定を変更することを特徴とする請求項 6 記載の共有ライブラリ呼び出し方法。

【請求項 8】

前記出力アプリケーションが前記監査のポリシーを設定するコマンドを発行し、

前記コマンドを、前記カーネル領域の所定の領域に格納し、

前記モジュールが前記所定の領域に格納されたコマンドに基づいて前記監査のポリシーを設定することを特徴とする請求項 7 記載の共有ライブラリ呼び出し方法。

【書類名】 明細書**【発明の名称】 共有ライブラリに格納された外部公開関数の呼び出し方法****【技術分野】****【0001】**

本発明は、計算機で実行されるアプリケーションによる共有ライブラリに格納された外部公開関数の呼び出し方法及びその方法を実行する計算機に関する。

【背景技術】**【0002】**

一般の計算機は、CPU、メモリ、ストレージ、通信装置、キーボード、マウス又はカード読取装置といった入力装置、ディスプレイ又はプリンタといった出力装置で構成される。

【0003】

計算機上で動作するソフトウェア、具体的にはオペレーティングシステム（以下「OS」）、アプリケーション、共有ライブラリ等は、ストレージに格納されている。

【0004】

尚、共有ライブラリとは、ストレージ103の記憶容量又はメモリ105の使用量を削減するため、ソフトウェア世代管理手続きを削減するため、又はソフトウェア開発コストを削減するために、種々のプログラムで実行される定型的な手続き（関数等）を集めたファイルあるいはモジュールである。この共有ライブラリは、複数のプログラムで共有されるため、アプリケーション等のプログラム本体とは別個に作成、管理、保存される。

【0005】

共有ライブラリには、プログラムコンパイル時にプログラム本体と結合される静的結合ライブラリと、実行時に結合される動的結合ライブラリとがある。

【0006】

計算機では、図4に示すように、アプリケーションが使用する記憶空間を、アプリケーション領域1（401）、アプリケーション領域2（402）、アプリケーション領域3（404）のようにアプリケーション固有に持たせる場合が多い。ただし、全ての処理に共通して使用されるOSについては、アプリケーションとは別の記憶領域が割り当てられる。この領域をカーネル領域407という。通常、このカーネル領域407は、特別な権限が無いアプリケーションからは自由にアクセスされない。

【0007】

これによって、各アプリケーションは他のアプリケーションが含む問題によって干渉されたり、破壊されたりすることを防ぐことができる。このとき、動的結合ライブラリは、OSの処理によって、実行コード領域とプログラム実行中に変更されない定数領域が、共有ライブラリ領域403、405、406のように、複数のプロセス間で共有される、即ち、一つの共有ライブラリが複数のメモリ領域にマッピングされるようになっている場合が多く、この場合、メモリ使用量削減やプログラム起動時間の削減効果が期待できる。

【0008】

また、アプリケーションからカーネル領域407へのアクセスを制限しているOSの場合、一般的に、アプリケーションはカーネル領域407へのアクセス要求（以下「システムコール」）を発行する際、カーネル領域407で実行されているOS201等のプログラムに対して、ソフトウェア割り込みを発行する。

【0009】

ところが、OSによってはシステムコールをカーネル領域407へのソフトウェア割り込みとしてではなく、動的結合ライブラリとして実装している場合がある。具体的には、アプリケーションがカーネル領域407で実行されているOSを呼び出す場合、一旦共有ライブラリを呼び出し、その共有ライブラリ内でソフトウェア割り込みを発生させることで、カーネル領域407へアクセスするのである。

【0010】

非特許文献1には、こうしたアプリケーションからの共有ライブラリの呼び出しを、割

り込みに関わる OS やアプリケーションと異なるプログラム（以下「外部プログラム」）によって捕捉（以下「フック」）し、その呼出内容を記録し、又はフィルタリングする方法として動的結合ライブラリへの注入という技術が開示されている。

【0011】

注入は、図5に示すように、注入対象、すなわち呼び出しの対象となる共有ライブラリ（以下「注入対象共有ライブラリ」）503と同じ外部公開関数を有する注入共有ライブラリ502を作成し、注入対象共有ライブラリの代わりにアプリケーション501と結合（アプリケーションの記憶空間にマッピング）させる。さらに注入共有ライブラリは、アプリケーションの共有ライブラリ呼出に対する、記録、監査、拒否といった処理を行う。その処理が終了した後、注入共有ライブラリは、注入対象である共有ライブラリを呼び出すよう設計されるので、アプリケーションからはシステムが注入共有ライブラリなしで運用されている場合と同じように動作しているように見える。

【0012】

【非特許文献1】Jeffrey Richter著 長尾高弘訳 Advanced Windows 改訂第3版 アスキー出版局。

【発明の開示】

【発明が解決しようとする課題】

【0013】

従来技術では、共有ライブラリに対応する記憶領域403等が各アプリケーションの記憶領域401等の中に配置されるため、共有ライブラリから呼び出された関数等の変数が格納される記憶領域はアプリケーション毎の個別の記憶領域になるという問題がある。例えば、注入共有ライブラリによって共有ライブラリ呼び出しの記録をする場合、記録すべき変数等の情報が各アプリケーションに割り当てられた記憶空間に配置されていると、その情報を収集するために、計算機ではプロセス間通信や記憶空間切り替えを行って、目的の情報を有する記憶空間に仮想記憶空間を切り替える必要が発生し性能の低下を招く。またプログラムが複雑になることによって信頼性の低下を招く。

【0014】

上記の不都合を解消するために、アプリケーションが配置されている記憶空間に各アプリケーションが共通に使用できる領域を設け、共有ライブラリで使用される変数等の情報を記録することが考えられる。しかし、時間の経過によって実行されるアプリケーションが切り替えられる可能性がある。この場合、一貫性をもった記録を作成するには、アプリケーション間でのロック機構を設ける必要がある。これは記録を頻繁に残さなければならない場合に性能低下につながり、また複数のロックを取得しなければならない場合、デッドロックに陥ってシステムが停止してしまう危険が発生するという問題がある。

【0015】

さらに、共有ライブラリの動作を制御（変数を指定）することを考えた場合、共有ライブラリの変数領域はアプリケーション毎に個別に用意されるため、ファイル等共有領域を設けて、そこに動作制御用のコマンドを書き込み、共有ライブラリが定期的に共有領域を参照することによって、動作制御用のコマンドを受信するという構成にする必要がある。しかし、ファイル等の共有領域の読み込みは、システムコールに比べて非常に時間がかかるので、性能上問題があるうえ、システムの状態によってはファイルアクセスができない場合があるという問題もある。

【課題を解決するための手段】

【0016】

本発明では、上記課題を解決するため、アプリケーションから共有ライブラリに格納された外部公開関数の呼び出しを注入共有ライブラリでフックする際に使用する注入共有ライブラリの共有領域や共有ライブラリへのコマンド伝達領域をカーネル領域に設ける。

【0017】

又、注入共有ライブラリで外部公開関数の呼び出しを監査する際に使用される基準についての情報を格納する領域もカーネル領域に設ける。

【0018】

更に、各アプリケーションの共有ライブラリからの外部公開関数の呼び出し履歴（以下「ログ」）を格納する領域も、カーネル領域に設ける。

【0019】

本発明の他の特徴については、本明細書及び添付図面の記載により明らかにする。

【発明の効果】

【0020】

共有ライブラリの外部公開関数へのアクセスを共有ライブラリを書換え無しにフックしてフィルタリングする際、一貫性のあるログを取得することが可能となる。

【発明を実施するための最良の形態】

【0021】

以下、実施形態の概要について説明する。

本実施形態においては、計算機101は、ソフトウェアプログラムとして、共有ライブラリ、共有ライブラリと関連付けられ、アプリケーションに結合される注入共有ライブラリ、カーネル領域に配置され、注入共有ライブラリがログ記録、管理者からのコマンド受信、共有ライブラリ外部公開関数呼び出し監査などの処理を行うフィルタモジュール及び注入共有ライブラリがカーネル領域に設けられた共通領域（以下「カーネルモード共有領域」）へ出力した結果を収集し、ファイルへの出力、管理者への報告などを行う出力アプリケーションを備える。

【0022】

注入共有ライブラリは、注入対象となる共有ライブラリが有する外部公開関数へのアクセスの一部または全部をフックする。

【0023】

カーネルモード共有領域は、カーネル領域の拡張であっても、カーネルへの動的追加モジュールであっても、カーネルモードで動作するデバイスドライバで実現されても良い。また、カーネルモード共有領域内には共有ライブラリ呼び出しの監査に使用する監査ポリシが配置される場合もある。

【0024】

出力アプリケーションは、定期的またはカーネルモード共有領域からの呼び出しに基づいて、カーネルモード共有領域内に記録された情報を特定のファイルに出力する、管理者に報告を行うといった処理を行う。また、システム管理者からの指示といった注入共有ライブラリへのコマンドは、出力アプリケーションがカーネルモード共有領域内のコマンド受信領域に対し、書き込みを行うことによって発信することとできる。

【0025】

アプリケーションから共有ライブラリへの関数の呼び出しが発生すると、呼び出しは注入共有ライブラリにフックされ、注入共有ライブラリへと制御が移る。注入共有ライブラリでは、呼び出し時刻、呼び出しを行ったユーザ、呼び出しを行ったアプリケーション、呼び出し時に渡された引数といった情報を取得することが可能である。注入共有ライブラリでは、これらの情報のカーネルモード共有領域への記録を、カーネル領域で実行されているフィルタモジュールにシステムコールの形で依頼する。フィルタモジュールは、受信した情報を記録し、監査基準に照らして、正当なアクセスであるかを監査する。この際、フィルタモジュールがコマンド受信領域を読み込んで、ログ取得実施やアクセス制御の有無を制御することができる。監査基準に照らして、不正であると判断されたアクセスについては、本来の共有ライブラリへの呼出を行わずに、記録を残すのみとする、いわゆる共有ライブラリへのアクセスのフィルタリングを行うことができる。

【0026】

以下、実施形態の詳細について説明する。

【0027】

図1は、本実施形態における計算機のブロック例を示す図である。計算機101は、CPU102、メモリ105、ストレージ103、通信装置106、キーボード、マウス又

はカード読取装置といった入力装置104、ディスプレイ又はプリンタといった出力装置107を有する。

【0028】

図3に示すように、計算機101上で動作するソフトウェア、具体的にはOS301、アプリケーション302、共有ライブラリ303等は、ストレージ103に格納されている。アプリケーション302や共有ライブラリ303は複数存在する場合が多い。ここで、ストレージ103は、ハードディスクドライブやDVDドライブといった、不揮発性の記憶装置である。

【0029】

これらのソフトウェアを実行する際、計算機101は、図2に示すように、ストレージ103よりOS301、アプリケーション302及び共有ライブラリ303を揮発性だが高速なメモリ105上に、OS201、アプリケーション202及び共有ライブラリ203としてロードしてCPU102で実行する。以下、動作主体をソフトウェアで説明するが、実際は、そのソフトウェアを実行するCPU102によって動作は実行される。また、以下で説明されるソフトウェア間の通信は、指定された記憶領域にコマンドを記録することによるプログラム間通信によって行われる。

【0030】

図6は、本実施形態における計算機101が有するソフトウェアのソフトウェア構成を示す図である。通常のアプリケーションプログラムは、アプリケーションが使用できる記憶領域（以下「ユーザモード領域」）601内に、アプリケーション1（603）やアプリケーション2（606）のように配置される。アプリケーションプログラム603、606は、コンパイル時または実行時に注入共有ライブラリ604、607と結合される。

【0031】

注入共有ライブラリ604、607は、共有ライブラリ（605、608）とそれぞれ結合される。また、出力アプリケーション609も、ユーザモード領域601内に配置される。

ログ用共有領域610、コマンド受信領域611及び監査ポリシ612を有するフィルタモジュール613は、カーネル領域602に配置される。

【0032】

アプリケーション1（603）、注入共有ライブラリ604、共有ライブラリ605及びフィルタモジュール613は同一の仮想空間に配置される。同様に、アプリケーション2（606）、注入ライブラリ607、共有ライブラリ608及びフィルタモジュール613は同一の仮想空間に、出力アプリケーション609とフィルタモジュール613は同一の仮想空間にそれぞれマップされる。アプリケーション1、アプリケーション2、出力アプリケーションは同時にマップされることはない。

【0033】

アプリケーション1（603）およびアプリケーション2（606）はそれぞれ、注入対象となる共有ライブラリ605、608を呼び出す際には、図7に示す手順に従う。以下、図7に示された手順について説明する。尚、以下の説明では、アプリケーション1についてのみ説明するが、他のアプリケーションでも同様である。

【0034】

まず、アプリケーション1は、共有ライブラリ605の外部公開関数を呼び出す（ステップ701）。

共有ライブラリ605への外部公開関数の呼び出しを受けたら、注入共有ライブラリ604は、その共有ライブラリ605への呼び出しをフックする。具体的なフック方法として、コンパイル時に注入共有ライブラリを静的にリンクする方法、動的リンクライブラリを注入ライブラリで置き換える方法、他アプリケーション空間で実行されているアプリケーションを用いて、目的とする外部公開関数へのアクセスをフックする方法などがある（ステップ702）。

【0035】

【 0 0 3 6 】

【 0 0 3 7 】

【 0 0 3 8 】

【 0 0 3 9 】

【 0 0 4 0 】

【 0 0 4 1 】

【 0 0 4 2 】

【 0 0 4 3 】

出証特 2 0 0 3 - 3 1 0 9 6 8 6

5の外部公開関数を、適切な引数を伴って呼出すことが可能である。

【0044】

共有ライブラリ605から外部公開関数を呼び出した共有注入ライブラリ604は、その結果をアプリケーションに送信する。

【0045】

共有ライブラリ呼出の監査の結果、呼出が許可されなかった場合、フィルタモジュール613は、ログ用共有領域610に不許可となった呼出の種類、不許可となった理由などを記録した上でアプリケーションにエラーを返す。エラーを返す他の方法として、共有ライブラリ605の呼出は行わず、アプリケーションに正常終了を返す方法、フィルタモジュール613が、許可されなかった呼出を共有ライブラリ605への呼出を許可される形に補正、すなわち呼出に使用される引数を許可される値に変更するといった補正を行ったうえで、注入共有ライブラリが共有ライブラリ呼出を行う方法などがある（ステップ711）。

【0046】

図8は、ステップ703で、注入共有ライブラリがフィルタモジュールに対して送信する共有ライブラリ呼出情報の例を示す図である。共有ライブラリ呼出情報には、共有ライブラリ呼出元であるアプリケーション603の名前を示すアプリケーション名801、アプリケーション603のストレージ103に作成されたファイルシステム上の格納位置を示すアプリケーションパス802、アプリケーションを実行しているユーザを示すユーザ803、アプリケーションを実行しているユーザの所属するグループを示すグループ804、共有ライブラリ呼出時刻を示す呼出時刻805、アプリケーション内呼出位置を示す呼出元アドレス806、アプリケーションが共有ライブラリ呼出の際渡した引数の数を示す引数の個数807及びアプリケーションが共有ライブラリ呼出の際渡した引数の内容を示す引数のリスト808が含まれる。

【0047】

図9は、監査ポリシ612に格納され、共有ライブラリ呼出の監査に用いられる監査ポリシの一例を示す図である。この監査ポリシは、少なくとも注入対象となる共有ライブラリの数だけ用意される。

【0048】

列901は、監査対象となる共有ライブラリの外部公開関数名が登録される列である。以下の列に指定される項目は、列901で指定される関数への呼出を許可する条件を構成する。

【0049】

列902は、列901で指定された関数への呼出を許可するアプリケーションの名前が登録される列である。登録される情報として、特定のアプリケーション名の他に、アプリケーション群を示す記号などを用いることで、アクセスを許可するアプリケーションの表記を簡略化する場合もある。

【0050】

列903は、列901で指定された関数への呼出が許可されるアプリケーションが保存されるファイルシステム上の位置を表すアプリケーションパスが登録される列である。列902に登録された情報と組み合わせて使用することで、特定のパスに配置された任意のアプリケーションを指定し、アクセスを許可をすることができる。

【0051】

列904は、列901で指定された関数への呼出が許可されるユーザを示す情報が登録される列である。登録される情報として、特定のユーザ名の他に、ユーザ郡を示す記号などを用いることで、アクセスを許可するユーザの表記を簡略化する場合もある。

【0052】

列905は、列901で指定された関数への呼出を許可するユーザが所属するグループを示す情報が登録される列である。列904に登録された情報と組み合わせることによって、特定グループに所属する任意のユーザといったアクセス許可の仕方が可能となる。

【0053】

列906は、列901で指定した関数への呼出が許可される時刻や時間帯を示す情報が登録される列である。

【0054】

列907は、列901で指定した関数への呼出が許可されるアドレスを示す情報が登録される列である。特定のアドレス範囲内からの呼出を受け付けなくする指定や、コード領域からの呼出のみ許可する、スタック領域からの呼出を不許可とするといった指定が可能である。

【0055】

列908は、列901で指定された関数への呼出を監査する際に、引数の内容まで監査する場合の監査方法が登録される列である。例えば、第1引数に80という値が入っていなかった場合、アクセスを不許可とするといった監査内容が記載される。監査方法の他の実施形態として、アクセス監査モジュールを外付け、即ち、注入共有ライブラリ604にアクセス監査モジュールを機能的に追加できるようにしておき、監査ポリシーの列908でアクセス監査モジュールの指定を行うといった方法もある。

尚、監査条件は、必要に応じて追加されても良い。

【0056】

監査ポリシーを用いた監査では、901で指定された関数に対し、902～908で示された監査条件をすべて満たした場合のみ、アプリケーションの共有ライブラリの呼出が許可される。

【0057】

本実施形態の計算機では、監査ポリシー612等のフィルタモジュール613の設定の変更が出力アプリケーション609を用いて行われる。具体的には、フィルタモジュール613の挙動を変更する場合、管理者は出力アプリケーション609を用い、コマンド受信領域611へ変更すべき情報の書き込みを行う。たとえばシステム運用中に、監査ポリシー612を更新する場合の手順について、図10を用いて説明する。

【0058】

管理者は監査ポリシー変更の指示を出力アプリケーション609に入力する（ステップ1001）。管理者から監査ポリシー変更指示を受けた出力アプリケーション609は、フィルタモジュール613のコマンド受信領域611に対し、監査ポリシー変更コマンドを出力する（ステップ1002）。その後、フィルタモジュール613は出力アプリケーション609によって出力された管理ポリシー変更コマンドをコマンド受信領域611から読み取り、コマンドに従って監査ポリシー612を変更する（ステップ1003）。

【0059】

尚、管理者は、上記で示した管理ポリシー612の変更の指示の他にも、フィルタモジュール613の動作開始や停止などの指示を出力アプリケーション609を用いて行うことができる。

【0060】

次に、本実施形態を計算機のセキュリティを向上するソフトウェアに適用した例について説明する。

図11は、カーネル領域602にセキュリティモジュールを追加して、セキュリティを向上させるソフトウェアに対し、本実施形態を適用した場合のソフトウェア構成例を示すブロック図である。本来、セキュリティモジュール1107はセキュリティを保つためカーネル領域602に常駐している必要がある。しかし、セキュリティモジュール1107は、カーネル領域602内に存在するためには一般的なカーネルモジュールと同じ機能を備える必要があるため、悪意あるアプリケーションプログラム1103が適切な権限を取得することができれば、共有ライブラリ605を呼び出して、セキュリティモジュール1107をアンロードするまたは不活性化するという無効化することができるという問題点がある。

【0061】

しかし、図 11 に示すように本実施形態を適用することで、矢印 1108 に示すように、悪意あるアプリケーション 1103 が共有ライブラリ 604 にアクセスする前に注入共有ライブラリ 604 によって、フィルタリングモジュール 607 にアクセスが通知される。更に、フィルタリングモジュール 607 が、自身に設定されている監視ポリシーに基づいて、セキュリティモジュール 1107 の無効化を許可しないように動作する。これにより、意図しないセキュリティモジュールの無効化を防ぐことができる。

【0062】

上述したように、フィルタモジュールをカーネル領域に設けることで、注入共有ライブラリからの共有ライブラリへのアクセス要求を監査する等の要求が、フィルタモジュールに対する一種のシステムコールになるので、計算機にロック機構を導入する必要が無い。又、カーネル領域においてアプリケーション毎にログを管理するので、処理情報の一貫性を保つことができる。又、ログ情報の取得の際にシステムコールを用いることで、処理速度が向上する。

【図面の簡単な説明】

【0063】

【図 1】 計算機のハードウェア構成の例を示す図である。

【図 2】 メモリの内容例を示す図である。

【図 3】 ストレージの内容例を示す図である。

【図 4】 計算機上の仮想空間レイアウト例を示す図である。

【図 5】 計算機上の一仮想空間上のモジュール配置例を示す図である。

【図 6】 計算機上の仮想空間レイアウト例を示す図である。

【図 7】 共有ライブラリ呼出ログ取得および共有ライブラリ呼出監査手順をしめすフローチャートである。

【図 8】 共有ライブラリ呼出情報例を示す図である。

【図 9】 監査ポリシー例を示す図である。

【図 10】 出力アプリケーションが、コマンド受信領域に監査ポリシー変更コマンドを出力する手順をしめすフローチャートである。

【図 11】 計算機上の仮想空間レイアウト例を示す図である。

【符号の説明】

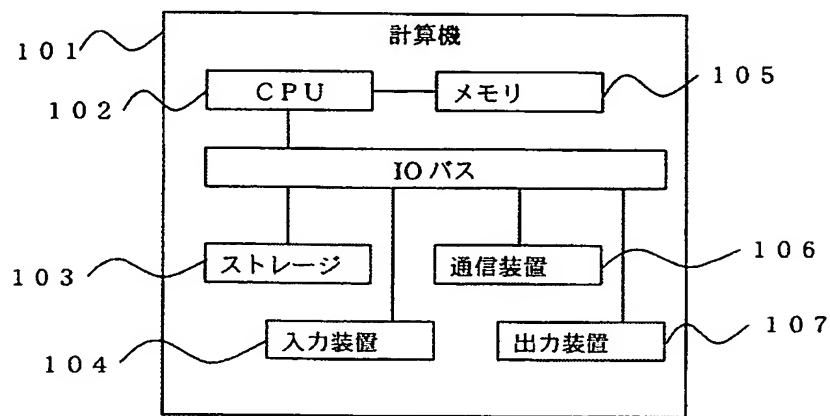
【0064】

101…計算機、102…CPU、103…ストレージ、104…入力装置、105…メモリ、106…通信装置、107…出力装置。

【書類名】 図面

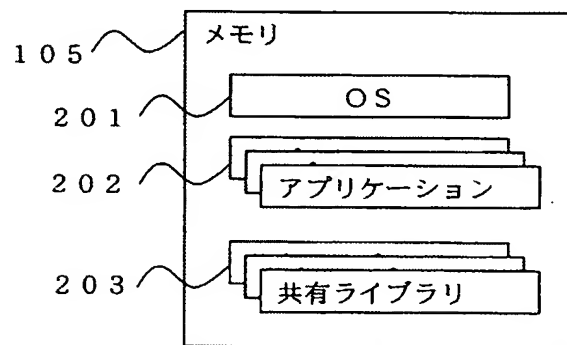
【図 1】

図 1



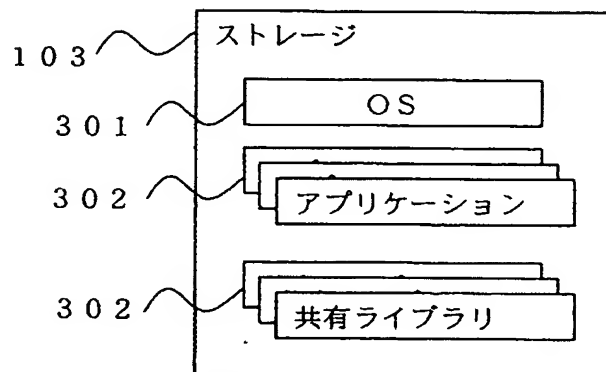
【図 2】

図 2

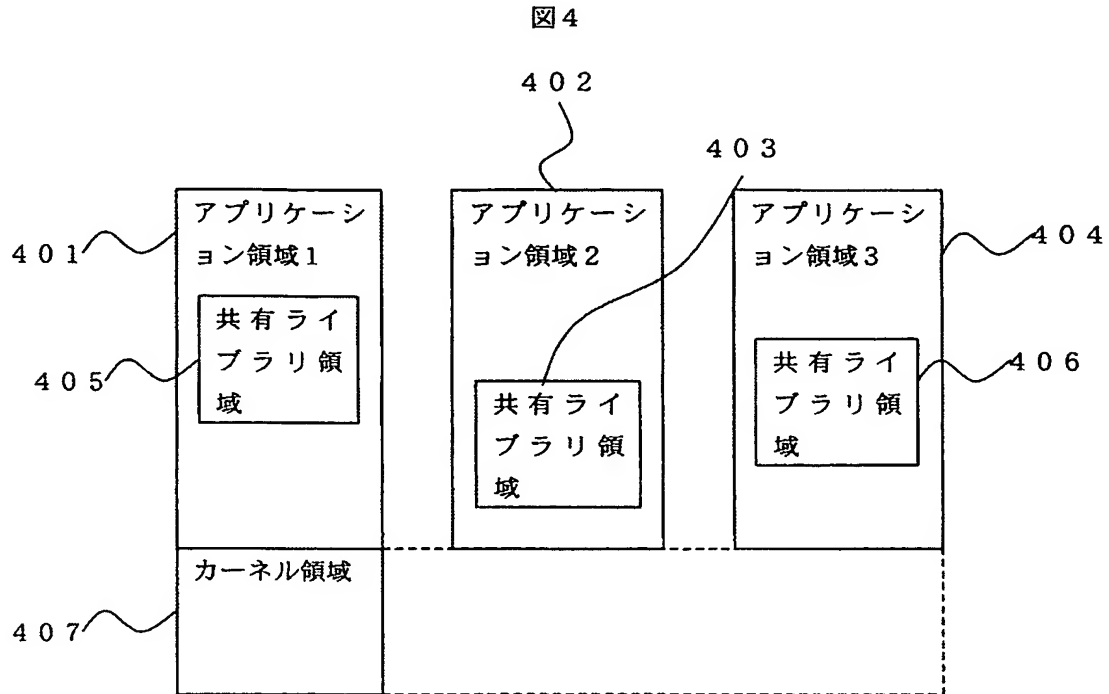


【図 3】

図 3

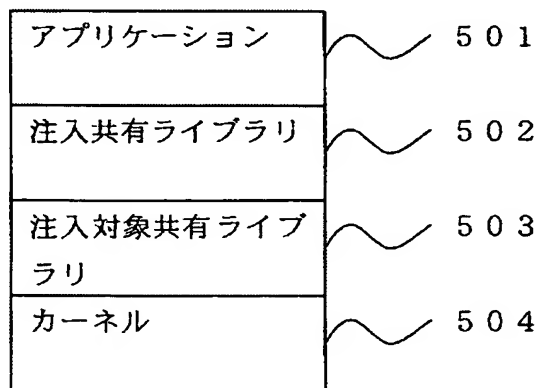


【図 4】



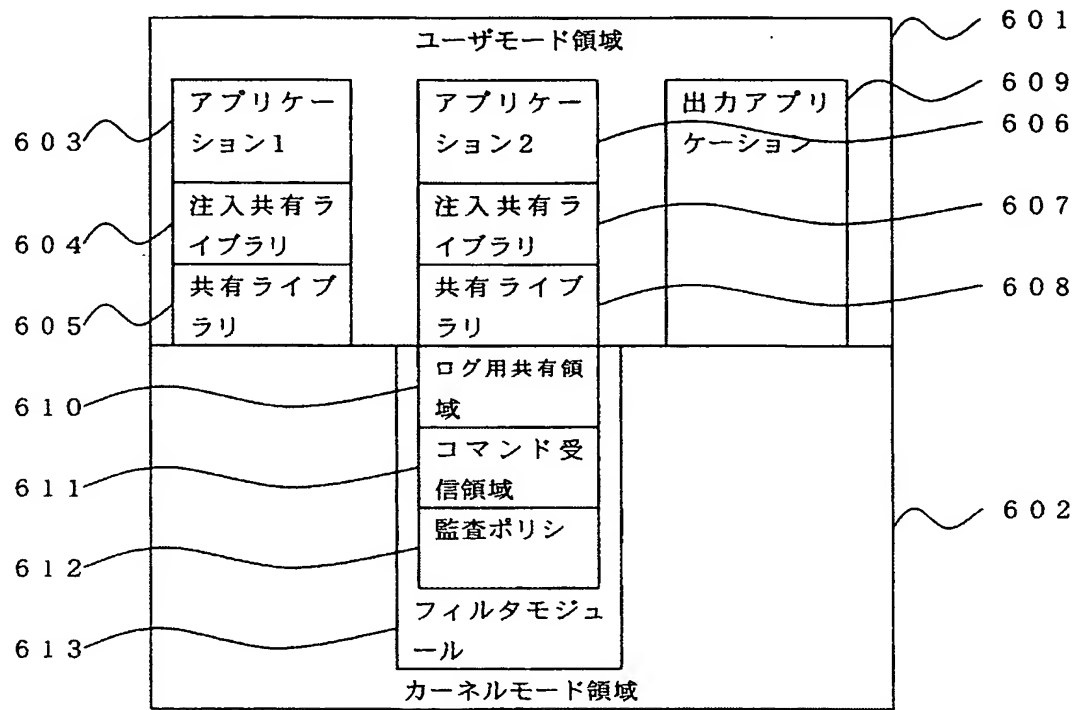
【図 5】

図 5



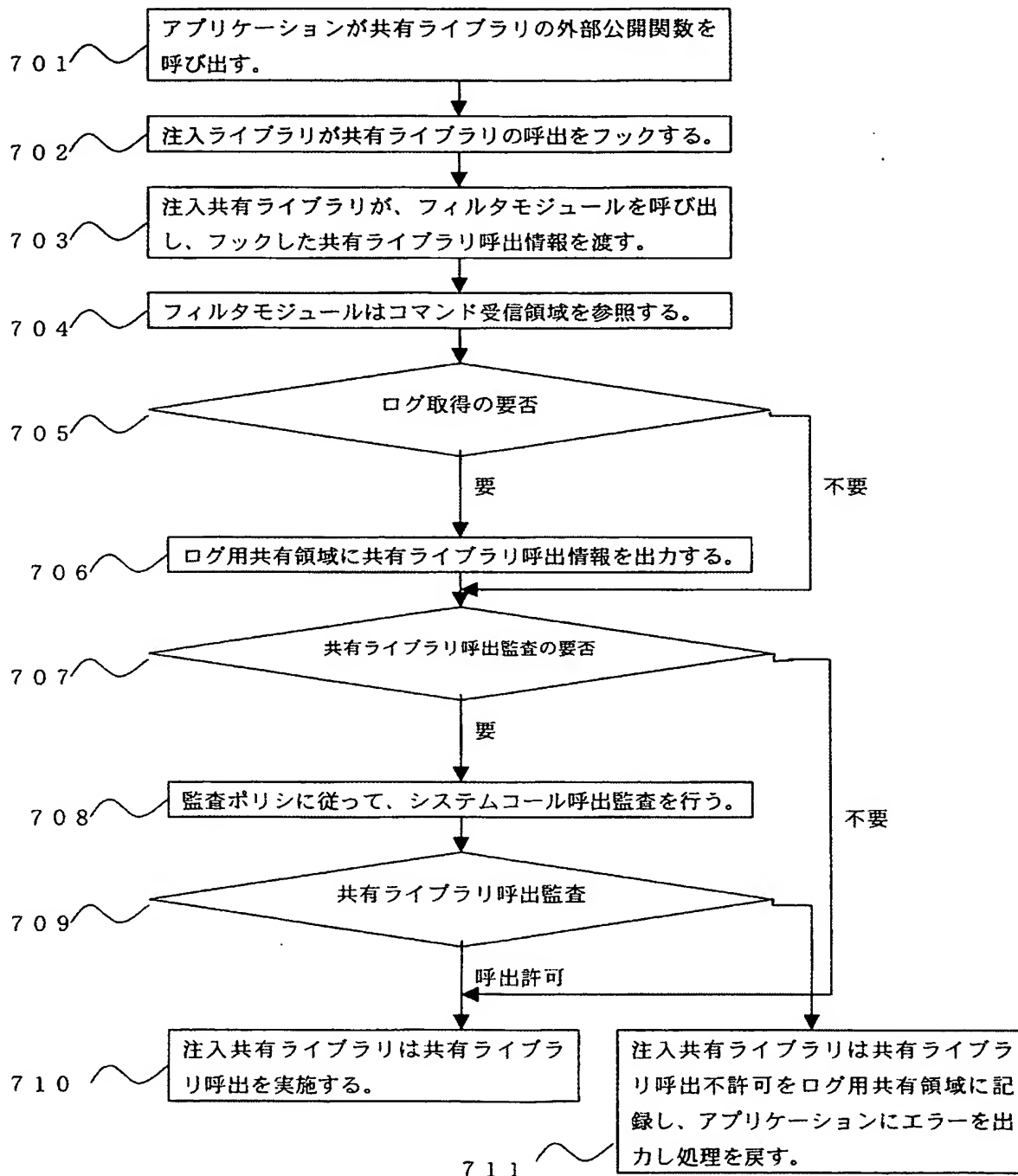
【図 6】

図 6



【図 7】

図 7



【図 8】

図 8

アプリケーション名	801
アプリケーションパス	802
ユーザ	803
グループ	804
呼出時刻	805
呼出元アドレス	806
引数の個数	807
引数のリスト	808

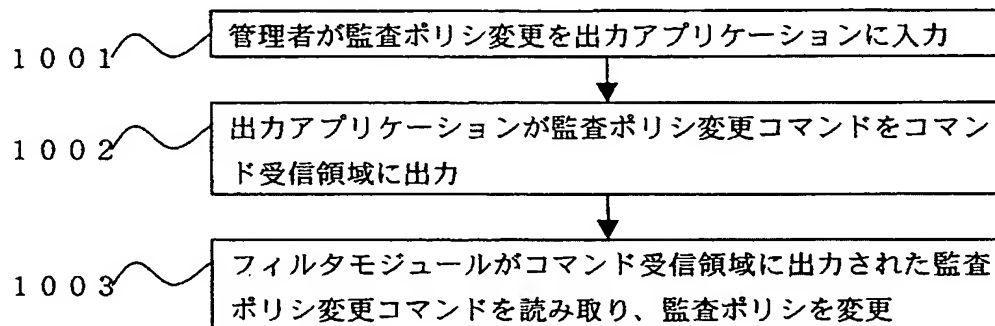
【図 9】

図 9

関数名	アプリケーション名	アプリケーションパス	ユーザ	グループ	許可時刻	許可アドレス	引数監査内容	909
open	App1	/bin/	User1	Grp1	8:00-17:00	7000000		910
open	App2	/usr/bin/	User2	Grp2	24h	2000-4000		911
close	App3	/var/bin/	User3	Grp3	17:00-0:00	80000000		912
								...

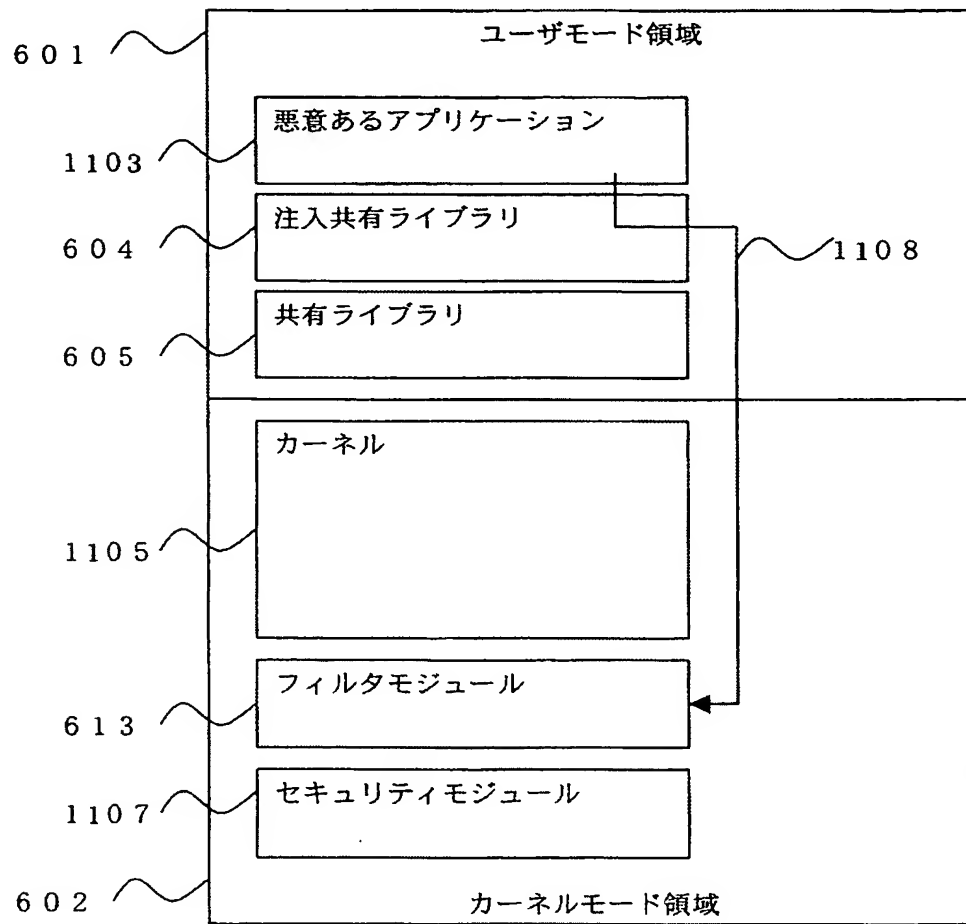
【図 10】

図 10



【図 11】

図 11



【書類名】 要約書

【要約】

【課題】

共有ライブラリを介した外部公開関数への呼び出しを効率よく監査したい。

【解決手段】

共有ライブラリの外部公開関数をフックする注入共有ライブラリを用いてアプリケーションの共有ライブラリ呼出をフックし、カーネル領域に配置したフィルタモジュールで、共有ライブラリ呼出情報の記録、共有ライブラリ呼出の監査を行うことによって、共有ライブラリの変更無しに共有ライブラリ呼出をフィルタリングし、高速に一貫性のある記録を作成する。

【選択図】 図6

認定・付加情報

特許出願の番号	特願 2003-386091
受付番号	50301891726
書類名	特許願
担当官	第七担当上席 0096
作成日	平成15年11月18日

＜認定情報・付加情報＞

【提出日】	平成15年11月17日
-------	-------------

特願 2003-386091

出願人履歴情報

識別番号

[000005108]

1. 変更年月日

1990年 8月31日

[変更理由]

新規登録

住 所

東京都千代田区神田駿河台4丁目6番地

氏 名

株式会社日立製作所